The History and Future of Account Abstraction

Ismael Darwish

Follow

Nethermind.eth

--

Listen

Share

By

[Ismael Darwish

](https://twitter.com/ismael_eth)with help from

[Kristof Gazso

](https://twitter.com/kristofgazso),

[Yoav Weiss

](https://twitter.com/yoavw)and

[Liraz Siri

](https://twitter.com/lirazsiri)

Ethereum today has accomplished many milestones, and with its rise in popularity, more and more people are starting to dabble into the blockchain world. But if we want to achieve mainstream adoption, a few significant issues need to be addressed:

- We need social recovery wallets
- Support multi-operation transactions
- Pay fees in currencies other than ETH

All these issues can be solved by EIP 4337, introducing Account Abstraction (AA) in Ethereum without requiring consensus-layer protocol changes. As you'll find out below, this can radically change how Ethereum is used and help onboard the next wave of users.

But first, what exactly is Account Abstraction?

Currently, in Ethereum there are two types of accounts: EOAs (or wallet accounts), controlled by the user's private key, and contract accounts, controlled by their code.

The goal of “account abstraction” is to reduce the number of account types from 2 (EOA and contract) to 1 (just contract) and to move functionality such as signature verification, gas payment and replay protection out of the core protocol and into the EVM.

This is currently not possible since all transactions in Ethereum today must start from an EOA, which has to have some ETH to pay for gas. This is hindering the ability to quickly onboard new users.

What do we get from it?

The key goal of account abstraction is to allow users to use smart contract wallets containing arbitrary verification logic. Smart contract wallets present many benefits over regular EOAs such as:

- Social recovery
- Multisig transactions
- Atomic multi-operation transactions

- More efficient, simpler, and quantum-safe signature algorithms
- Upgradeability

There are notably good solutions today, such as the Argent wallet, Gnosis safe, and Loopring that offer smart contract wallets with the current state of Ethereum. Still, they have drawbacks, such as the high cost to deploy the first time they're used and the need to use relayer services (often centralized) to submit the transactions. This leads to multiple different custom implementations to achieve a similar goal. By having a standard way for them to operate, we can solve those drawbacks.

If this is so important, how come it hasn't been implemented before?

Good question! Achieving account abstraction has been a long-time dream of many Ethereum developers. Let's dive into some of the formal proposals over the years and see how we got where we are today.

EIPs 86 and 1014 — First steps

Initially proposed by Vitalik Buterin in 2016, less than a year after Ethereum's initial release, EIP-86 tried to introduce smart contract wallets that could be thought of as "forwarding contracts". These would only accept transactions coming from an "entry point" address, an address from which anyone could send transactions given that they followed a specific format.

These forwarding contracts would be deployed to a specific address based on their code, introducing the idea that would later evolve to become EIP-1014, proposing the CREATE2 opcode.

The initial EIP-86 required significant changes to the protocol, and it ended up not being merged. It's important to note that changes to the protocol require coordination between node development teams and need to undergo extensive scrutiny since those rules must behave precisely in the same way in every Ethereum node. This makes the process of changing them very slow, and the vast majority of the proposals end up being stagnant or withdrawn.

Although EIP-86 didn't make it, EIP-1014 was merged in 2018, taking a huge step forward towards implementing easier to use smart contract wallets. Now a user could receive funds to a precalculated address even before deploying their smart contract wallet to it. The inclusion of CREATE2 is a crucial addition that would lay the foundation for some proposals we'll discuss in the coming sections.

EIP 2938 — A giant leap in the making

This EIP, proposed in September 2020 by Vitalik Buterin, Ansgar Dietrichs, and Matt Garnett, considers previous EIPs to introduce a new type of Ethereum transaction, an Account Abstraction transaction.

Special smart contracts identified as smart contract wallets would only accept this type of transaction. They would programmatically set the maximum gas for the transaction and implement an arbitrary verification method.

Although it's been developed for some time now, the nature of account abstraction poses many challenges.

EIP-2938 requires the addition of two new opcodes in the EVM to make the transaction viable. These opcodes change the core protocol significantly, and as stated in the previous section, the process of including such changes can be dragged for a long time.

In addition to this, there are still unanswered questions about how to achieve replay protection. Some possible solutions are suggested in EIP-2938, but there's still work to do to specify them properly.

On top of core protocol changes and uncertainty about replay protection solutions, there are more challenges regarding how nodes check the validity of these new types of transactions.

EIP-3074 — A tricky do-it-all

Created in October 2020 by Ansgar Dietrichs and Matt Garnett, among others, this EIP introduces two new opcodes: AUTH and AUTHCALL. When used together, they allow a smart contract to send a transaction on behalf of an EOA.

These opcodes are meant to be used in "invoker" contracts that can essentially supercharge any EOA.

The structure of the transaction that this contract can start is completely arbitrary, which makes it very easy to implement a wide array of solutions like multisig, batched, and sponsored transactions, key recovery, and more accessible CeFi exchange deposits.

This is a very versatile solution due to its unopinionated nature. However, it also poses some security risks. According to their proponents, these invoker contracts should undergo similar security audits to those for the beacon chain, since they're contracts that potentially all users of Ethereum would use.

EIP-3074 has received some criticism because of that, and Yoav Weiss proposed that making theAUTHCALL more opinionated would reduce the risks. Following these discussions, researchers started thinking of a better solution that would end up being proposed as EIP-4337, which we'll talk about later.

Apart from the security risks, the new opcodes would also modify the core protocol, which, as we've seen earlier, can delay the inclusion of a proposal.

L2s — The new frontier

As we've seen in the previous sections, introducing core protocol changes tends to be a big blocker in the process of introducing any EIP related to account abstraction. Core developers have been busy working on changes related to The Merge, which has been priority number 1 for quite a while now. But what about L2s?

Layer 2 chains that have been developed in the past years don't have the technical debt of Ethereum L1 and could introduce account abstraction out of the box. This is what many of them are doing:

- Optimism: This optimistic rollup uses its own OVM with opcodes that replace the EVM ones to allow full compatibility. In their first version they also implemented a basic form of Account abstraction by introducing three new functions in their execution manager that provide users with upgradeable smart contract wallets. Although very promising, they were removed in the second version in an effort to become identical to the EVM and for security concerns.
- StarkNet: The ZK rollup has its own implementation ofaccount abstraction.
- Argent, the L1 smart contract wallet has recently released the ArgentX, a version of their wallet on StarkNet using a custom implementation of account abstraction heavily inspired by EIP-4337.

These solutions highlight the importance of account abstraction and its relevance in any Ethereum blockchain.

But still, every project needs to come up with its own solution until we find a standard for them to operate on. Account abstraction will solve many issues but we can't have protocol changes anytime soon, and so far we haven't talked about any universal solution that could be implemented today.

Introducing EIP-4337: Account Abstraction in Ethereum without requiring consensus-layer protocol changes.

In September 2021, taking the learnings from previous efforts, Vitalik Buterin, alongside Ethereum researchers from OpenGSN and Nethermind proposed EIP-4337.

This EIP introduces account abstraction without any modifications to the core protocol. It achieves so by replicating the functionality of the transactions mempool in a higher-level system. However, instead of transactions, users send UserOperation objects to Ethereum nodes, and they package a set of these objects into a single transaction that gets included in the Ethereum chain.

This bundle transaction calls the "entry point" smart contract, which processes UserOperation objects and deploys smart contract wallets for them.

The deployed wallet completely handles nonces and signature verification. This offers a great deal of flexibility, allowing for smart contract wallets that include multisig and bundle transactions, social recovery, and ones that pay fees in ERC20 tokens.

As Vitalik himself stated, some sort of account abstraction will be implemented in Ethereum in the medium-term future. These solutions can first be included in new Layer 2 solutions and later make their way to Ethereum L1.

Using smart contract wallets by default will bring more and better possibilities for users to interact with Ethereum, and we could reach a point where we no longer need EOAs, and they get deprecated, relying entirely on the EVM for transactions. The new UserOperation mempool added in EIP-4337 could even replace the current transaction mempool altogether.

For a more detailed insight, you can check outVitalik's post and the ERC

This sounds amazing. When can we have it??

Good news, EIP-4337 is already here! We've started a collaboration that everyone is invited to join called Infinitism. This collaboration includes the teams at OpenGSN and Nethermind, who have been working hard on an implementation on the Goerli testnet that is already live. The Netheremind client already supports ERC-4337, and there's ongoing work to support it on Geth.

If you're interested, join the conversation over on our Discord!